

# **The List is Our Process!**

## **An analysis of the kernel's email-based development process**

Ralf Ramsauer\*, Sebastian Duda†, Lukas Bulwahn‡, Wolfgang Mauerer\*§

\* Technical University of Applied Sciences Regensburg

† Friedrich-Alexander University Erlangen-Nürnberg

‡ Hobbyist, active in LF ELISA Project, employed at BMW AG

§ Siemens AG, Corporate Research and Technology, Munich

Embedded Linux Conference Europe, Lyon

October 28, 2019



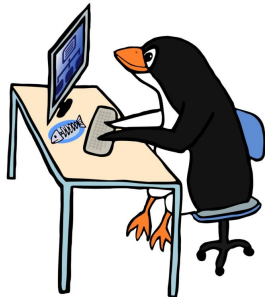
OSTBAYERISCHE  
TECHNISCHE HOCHSCHULE  
REGENSBURG



**SIEMENS**

## Our Overall Goal

Formalising and assessing the Linux  
Kernel development process



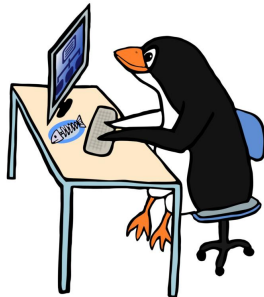
© Moini  
openclipart.org

## Our Overall Goal

Formalising and assessing the Linux  
Kernel development process

## Outside / Inside Motivation

- ▶ Safety-Critical Development
- ▶ Development Process Assessment
- ▶ Monitoring (cf. CHAOSS)
- ▶ Fundamentals of Software Engineering



© Moini  
openclipart.org

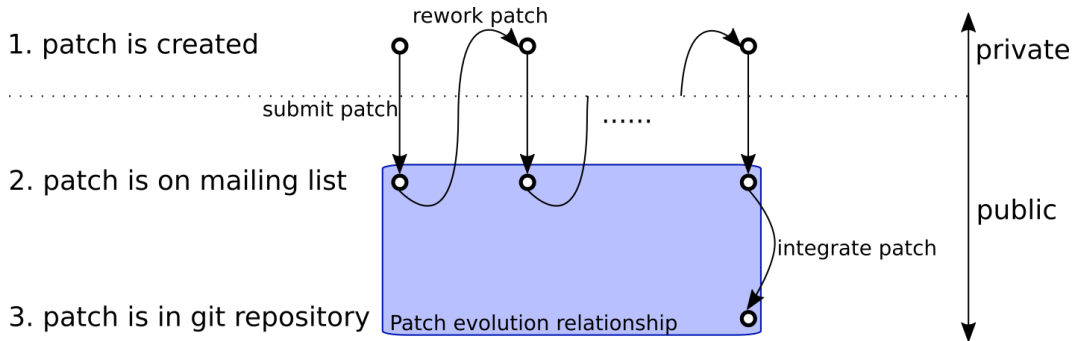
## Motivation from *inside* the community

### Interest of the kernel community itself

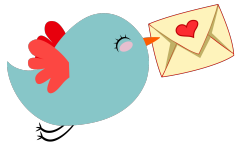
- ▶ D. Williams, Towards a Linux Kernel Maintainer Handbook, LPC 2018
- ▶ J. Corbet, Change IDs for kernel patches, <https://lwn.net/Articles/797613/>
- ▶ [Ksummit-discuss] [MAINTAINERS SUMMIT] Patch version changes in commit logs?
- ▶ [Ksummit-discuss] Allowing something Change-Id (or something like it) in kernel commits



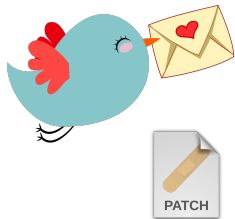
## Towards a formal model of the development process



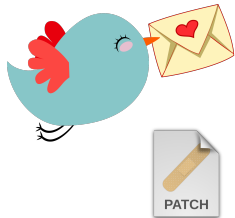
## Linux Kernel development workflow



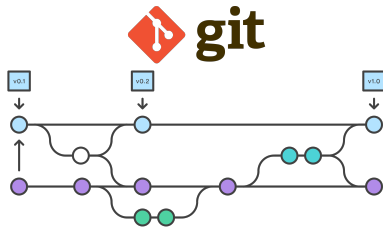
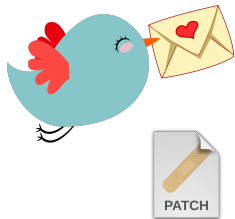
## Linux Kernel development workflow



## Linux Kernel development workflow



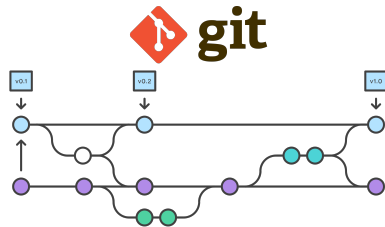
## Linux Kernel development workflow



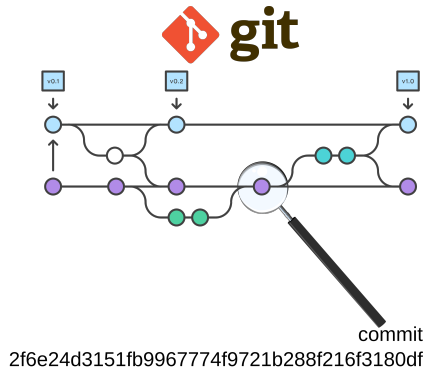
## Linux Kernel development workflow



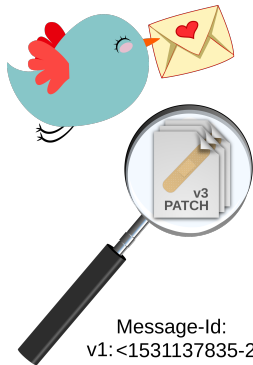
Message-Id:  
<1531137835-21581-1-git@1wt.eu>



## Linux Kernel development workflow

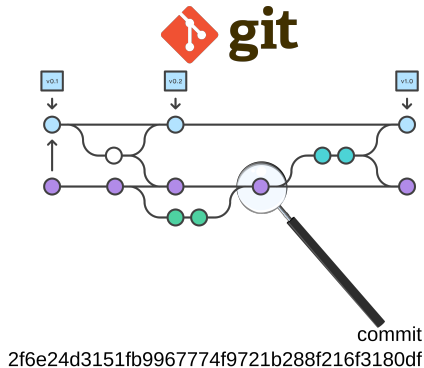


## Linux Kernel development workflow



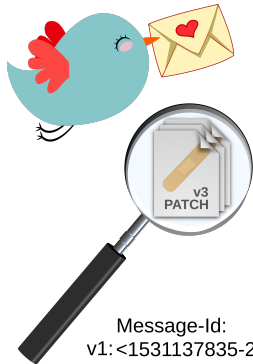
Message-Id:

v1: <1531137835-21581-1-git@1wt.eu>  
v2: <6739637657-68462-1-git@1wt.eu>  
v3: <9717683099-75474-1-git@1wt.eu>



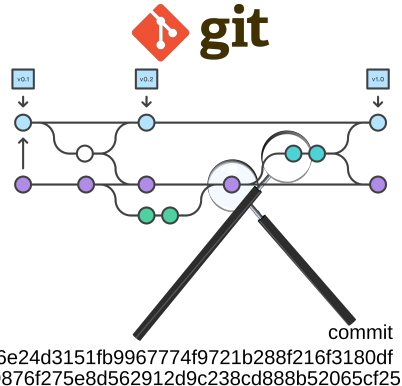


## Linux Kernel development workflow

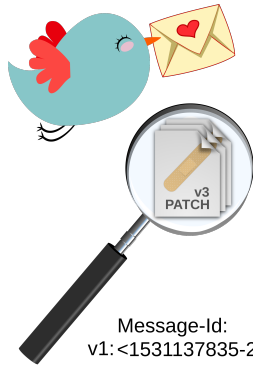


Message-Id:

v1: <1531137835-21581-1-git@1wt.eu>  
v2: <6739637657-68462-1-git@1wt.eu>  
v3: <9717683099-75474-1-git@1wt.eu>

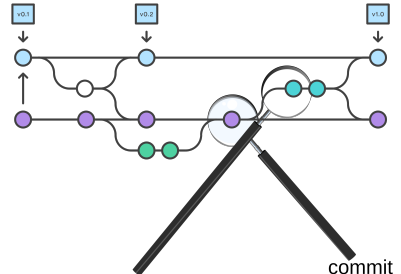


## Linux Kernel development workflow

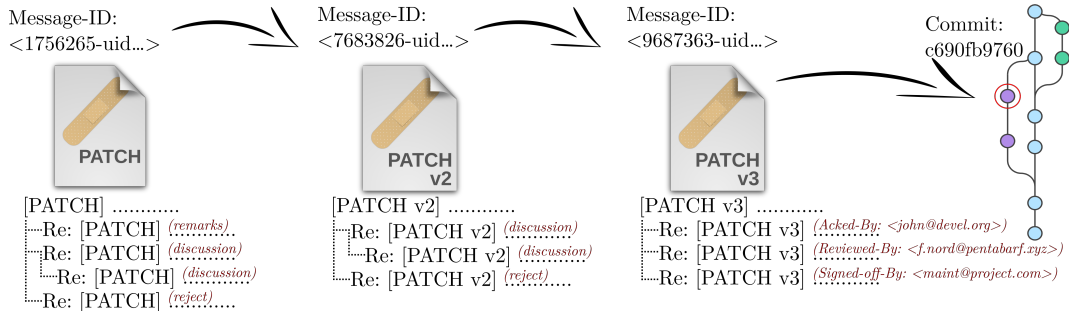


Message-Id:

v1: <1531137835-21581-1-git@1wt.eu>  
v2: <6739637657-68462-1-git@1wt.eu>  
v3: <9717683099-75474-1-git@1wt.eu>



2f6e24d3151fb9967774f9721b288f216f3180df  
89876f275e8d562912d9c238cd888b52065cf25



## PaStA - Patch Stack Analysis

- ▶ Detects *similar* patches across different branches
- ▶ Quantify mainlining efforts of off-tree developments (Preempt\_RT, vendor kernels, ...)
- ▶ Works with mailing lists!



Source: [toplock.net.au](http://toplock.net.au)

## Example of similar patches

```
commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date:   Mon Sep 12 21:45:49 2011 +0200
```

~~shed~~~~cure~~~~utter~~~~idle~~~~accounting~~~~madness~~.patch

Signed-off-by: Thomas Gleixner &lt;tglx@linutronix.de&gt;

```
diff --git a/kernel/sched.c b/kernel/sched.c
index 205499a..1121a97 100644
```

```

--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -5037,7 +5037,13 @@ EXPORT_SYMBOL(task_nice);
    */
    int idle_cpu(int cpu)
    {
-       return cpu_curr(cpu) == cpu_rq(cpu)->idle;
+       struct rq *rq = cpu_rq(cpu);
+
+       if (CONFIG_SMP)
+           return rq->curr == rq->idle && !rq->nr_running && !rq->wake_list;
+       else
+           return rq->curr == rq->idle && !rq->nr_running;
+       endif
    }

    /**

```



```
commit 908a3283728d92df36e0c7cd63304fd35e93a8a9
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Thu Sep 15 15:32:06 2011 +0200
```

```
sched: Fix idle cpu()
```

On -rt we observed hackbench waking all 400 tasks to a single cpu. This is because of `select_idle_sibling()`'s interaction with the new ipi based wakeup scheme.

[..snip..]

Signed-off-by: Thomas Gleixner <tglx@linutronix.de>  
Signed-off-by: Peter Zijlstra <a.p.zijlstra@chello.nl>  
Link: <http://lkml.kernel.org/n/tip-330p18b2> [...]  
Signed-off-by: Ingo Molnar <mingo@elte.hu>

```
diff --git a/kernel/sched.c b/kernel/sched.c
index 1874c74..4cdc91c 100644
```

```

--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -5138,7 +5138,20 @@ EXPORT_SYMBOL(task_nice);
 */
int idle_cpu(int cpu)
{
-   return cpu_curr(cpu) == cpu_rq(cpu)->idle;
+   struct rq *rq = cpu_rq(cpu);
+
+   if (rq->curr != rq->idle)
+       return 0;
+
+   if (rq->nr_running)
+       return 0;
+
+   #ifdef CONFIG_SMP
+   if (!list_empty(&rq->wake_list))
+       return 0;
+   #endif
+
+   return 1;
}

/**

```



# Example of similar patches

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b  
Author: Thomas Gleixner <tglx@linutronix.de>  
Date: Mon Sep 12 21:45:49 2011 +0200

~~sched-cure-utter-idle-accounting-madness.patrch~~

~~Signed-off-by: Thomas Gleixner <tglx@linutronix.de>~~

diff --git a/kernel/sched.c b/kernel/sched.c  
index 205499a..1121a97 100644

```
+ struct rq *rq = cpu_rq(cpu);  
+  
+ #ifdef CONFIG_SMP  
+ return rq->curr == rq->idle && !rq->nr_running;  
+ #else  
+ return rq->curr == rq->idle && !rq->nr_running;  
+ #endif
```

/\*\*



commit 908a3283728d92df36e0c7cd63304fd35e93a8a9  
Author: Thomas Gleixner <tglx@linutronix.de>  
Date: Thu Sep 15 15:32:06 2011 +0200

~~sched: Fix idle\_cpu()~~

On -rt we observed hackbench waking all 400 tasks to a single cpu. This is because of select\_idle\_sibling()'s interaction with the new ipi based wakeup scheme.

[...snip...]

~~Signed-off-by: Thomas Gleixner <tglx@linutronix.de>~~

```
+ struct rq *rq = cpu_rq(cpu);  
+  
+ if (rq->curr != rq->idle)  
+ return 0;  
+  
+ if (rq->nr_running)  
+ return 0;  
+ #ifdef CONFIG_SMP  
+ if (!list_empty(&rq->wake_list))  
+ return 0;  
+ #endif  
+  
+ return 1;  
+  
+ if (!list_empty(&rq->wake_list))  
+ return 0;  
+ #endif  
+ return 1;  
+ }  
+  
+ /**
```



# Example of similar patches

```

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Mon Aug 15 15:32:06 2011 +0200

    sched-cure: Fix idle_cpu()

Signed-off-by: Thomas Gleixner <tglx@linutronix.de>

diff --git a/kernel/sched.c b/kernel/sched.c
index 2054a1b..50377b2 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -5037,7 +5037,13 @@
    */
    int idle_cpu = 0;
    {
-       rq->idle = 1;
+       rq->curr = rq;
+       rq->idle = 0;
+       rq->nr_running = 0;
+       rq->wake_list = &rq->wake_list;
+       struct task_struct *p;
+       while ((p = rq->nr_running) != NULL)
+           return rq->curr == rq->idle && !rq->nr_running;
    }
}

/**

```



```

commit 908a3283728d92df36e0c7cd63304fd35e93a8a9
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Thu Sep 15 15:32:06 2011 +0200

```

sched: Fix idle\_cpu()

On -rt we observed hackbench waking all 400 tasks to a single cpu. This was due to the interaction of the idle\_cpu() function with the CONFIG\_SMP and CONFIG\_PREEMPT\_RT.

```

[...]
```

```

#endif
#ifdef CONFIG_SMP
    if (!list_empty(&rq->wake_list))
        return 0;
    rq->curr = rq;
    rq->idle = 0;
    rq->nr_running = 0;
    rq->wake_list = &rq->wake_list;
    struct task_struct *p;
    while ((p = rq->nr_running) != NULL)
        return rq->curr == rq->idle && !rq->nr_running;
}

return 0;

#ifdef CONFIG_SMP
    if (!list_empty(&rq->wake_list))
        return 0;
    return 1;
}

/**

```

```

*/
int idle_cpu(struct rq *rq)
{
    return 0;
}

#ifdef CONFIG_SMP
    if (!list_empty(&rq->wake_list))
        return 0;
    return 1;
}

/**

```



# Example of similar patches

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b

Author: Thomas Gleixner <tglx@linutronix.de>  
Date: Mon Sep 12 15:32:06 2011 +0200

```
diff --git a/sched-cpu.c b/sched-cpu.c
index 2054a3b..18b21b2 100644
--- a/sched-cpu.c
+++ b/sched-cpu.c
@@ -5037,7 +5037,13 @@
 */
int idle_cpu(int cpu)
{
-   return 0;
+   struct rq *rq;
+   rq->curr = rq->idle;
+   rq->nr_running = 0;
+   rq->wake_list = NULL;
+   return 1;
}

/**
```

commit 908a3283728d92df36e0c7cd63304fd35e93a8a9

Author: Thomas Gleixner <tglx@linutronix.de>

Date: Thu Sep 15 15:32:06 2011 +0200

sched: Fix idle\_cpu()

On -rt we observed hackbench waking all 400 tasks to a single

cpu. This was caused by the interaction of the idle\_cpu() function

with the CONFIG\_SMP patch. The patch was signed by Thomas Gleixner <tglx@linutronix.de> and the CONFIG\_SMP patch was signed by Linus Torvalds <torvalds@chello.nl>.

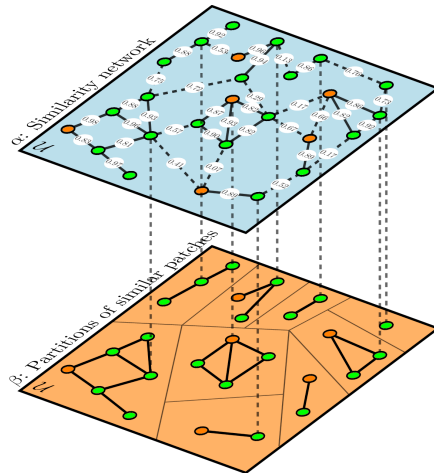
```
diff --git a/sched-cpu.c b/sched-cpu.c
index 18b21b2..513a3b2 100644
--- a/sched-cpu.c
+++ b/sched-cpu.c
@@ -513,7 +513,13 @@
 */
int idle_cpu(int cpu)
{
-   return 0;
+   struct rq *rq;
+   rq->curr = rq->idle;
+   rq->nr_running = 0;
+   rq->wake_list = NULL;
+   return 1;
}

/**
```

Diff similarity: 0.875







## Legend

- ▶ green nodes: patches on MLs
- ▶ orange nodes: commits in repository
- ▶ edges: similarity of patches/commits
  - ▶ dashed: similarity below thres
  - ▶ solid: similarity above thres

# Interested in the techniques? More details in:

## The List is the Process: Reliable Pre-Integration Tracking of Commits on Mailing Lists

Ralf Ramsauer<sup>1</sup>, Daniel Lohmann<sup>2</sup> and Wolfgang Mauerer<sup>3</sup>

<sup>1</sup>Technical University of Applied Sciences Regensburg  
University of Bamberg  
<sup>2</sup>Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

**Abstract**—A considerable corpus of research on software evolution focuses on mining changes in software repositories, but omits their pre-integration history.

We present a novel method for tracking this otherwise invisible evolution of software changes on mailing lists by connecting all early evidence of changes to their final version in repositories. Since artifact modifications on mailing lists are communicated by updates to fragments (i.e., patches only), identified semantically similar changes in a non-trivial task, we extract our method on high-profile open source software (OSS) projects like the Linux kernel, and validate its high accuracy using an elaborately created ground truth.

Our approach can be used to quantify properties of OSS development processes, which is an essential requirement for using OSS in safety-critical industrial products, where confidentiality and conformity in processes are critical. The high accuracy of our technique allows, to the best of our knowledge, for the first time to quantitatively determine if an open development process effectively aligns with given formal process requirements.

### 1. INTRODUCTION

Software patches may have come a long way before their final integration into the official branch (known as master or trunk) of a project. There are many possible ways of integration. Amongst others, the origin of a patch can be a merge from other developers' repositories (i.e., repositories of branches or patches, from foreign repositories), pull request on web-based repository managers such as GitHub or GitLab, vendor specific patch stacks, or mailing lists (MLs).

Especially MLs have been in use for software development processes for decades [17]. They have a well-known interface (plain text emails), and come with an established mixture of tool requirements (i.e., a mail user agent). Because of their simplicity, scalability, reliability and interface robustness, they are still widely used in many open source software (OSS) projects. In particular, mailing lists are a core infrastructure component of long-lasting OSS projects such as free-software systems (e.g., QEMU, Linux, GCC, etc.) operating

This work was supported by Siemens AG, Corporate Technology, the German Research Foundation (DFG) under grant no. LO 1174/1-1 (DFG) and the German Research Foundation (DFG) under grant no. LO 1174/1-1 (DFG). The work was supported by the BMBF under grant no. 01IS1401. The work was supported by the BMBF under grant no. 01IS1401. The work was supported by the BMBF under grant no. 01IS1401.

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

## Observing Custom Software Modifications: A Quantitative Approach of Tracking the Evolution of Patch Stacks

Ralf Ramsauer<sup>1</sup>, Daniel Lohmann<sup>2</sup> and Wolfgang Mauerer<sup>3</sup>  
Technical University of Applied Sciences Regensburg  
University of Bamberg  
Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

### ABSTRACT

Modifications to open-source software (OSS) are often provided in the form of "patch stacks" – sets of changes (patches) that modify a given body of source code. Maintaining patch stacks over extended periods of time is problematic, since the underlying base project changes frequently. This necessitates a continuous and engineering-intensive adaptation of the stack. Nonetheless, long-term maintenance is an important problem for changes that are not integrated into projects, for instance when they are controversial or only of value to a limited group of users.

We present and implement a methodology to systematically examine the temporal evolution of patch stacks, track non-functional properties like integrability and maintainability, and estimate the overall economic and engineering effort required to successfully develop and maintain patch stacks. Our results provide a basis for quantitative research on patch stacks, including statistical analyses and other methods that lead to actionable advice on the construction and long-term maintenance of custom extensions to OSS.

### 1. INTRODUCTION

Special-purpose software, like industrial control, medical analysis, or other domain-specific applications, is often composed of contributions from general-purpose projects that provide basic building blocks. These contributions are implemented on top of these built-in external dependencies, while the development of new ones, the primary branch of the base project, proceeds independently.

Especially for software with high dependability requirements, it is critical to keep up to date with analysis, latest from must be applied and new general features have to be introduced, as diverging software branches are hard to maintain and hard to integrate into the base system [8]. Possible development often ends in the form of patch stacks: Inter-granular modifications of existing code, which are not integrated into the base system, but are applied to the base system as a whole.

Permission to make digital or hard copies of all or part of this work for personal or commercial use, by registered users, is granted by ACM for non-profit organizations, provided that the fee code and the ACM copyright notice are included in the copy. For all other uses, permission should be sought from ACM. Copyright 2018 ACM 978-1-4503-5951-1/18/000000...\$15.00

October 30, August 17, 2018, Berlin, Germany  
© 2018 Copyright held by the owner(s). Publication rights licensed to ACM.  
10.1145/3211111.3211112

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

Siemens AG, Corporate Technology, Munich  
ralf.ramsauer@ofb.de, daniel.lohmann@siemens.de, wolfgang.mauerer@othr.de

## Data Acquisition

- ▶ Dumps from gmane.org etc.
- ▶ kernel.org public inboxes
  - ▶ some lists, prehistoric data
  - ▶ <https://lore.kernel.org/lists.html>
  - ▶ Some lists are imports from gmane.org :-)
- ▶ Our own collection



## Data Acquisition

- ▶ Dumps from gmane.org etc.
- ▶ kernel.org public inboxes
  - ▶ some lists, prehistoric data
  - ▶ <https://lore.kernel.org/lists.html>
  - ▶ Some lists are imports from gmane.org :-)
- ▶ Our own collection
  - ▶ 200 lists, since May '19
  - ▶ <https://github.com/linux-mailinglist-archives>



© Penguins of Madagascar  
20th Century Fox

## Let there be chaos

- ▶ Broken encoding
- ▶ BÅse64
- ▶
- ▶ MUAs
- ▶ Bots
- ▶ HTML
- ▶ Automated mails
- ▶ non-Linux patches
- ▶ Stable reviews
- ▶ Malformed recipients
- ▶ ...



Message-Id: <74851t0\$h3103wn0\$Delldi Fri, 9 Mar 71685 18:45:56  
+0000

Date: Mon, 08 Aug 05 04:01:15 ?x?\_???????

Date: Tue, 27 Mar 2001 13:42:39 +0200 (Westeuropäische  
Sommerzeit)



X-Mailer: Microsoft Outlook Express 6.00.2900.3028

## We analyse...

- ▶ v2.6.39..linus/master
- ▶  $\approx 610\text{K}$  commits
- ▶ Mails: 2011-05-01 - 2018-12-31
- ▶  $\approx 3\text{M}$  mails
- ▶ Lists: All Public Inboxes from [lore.kernel.org](https://lore.kernel.org)

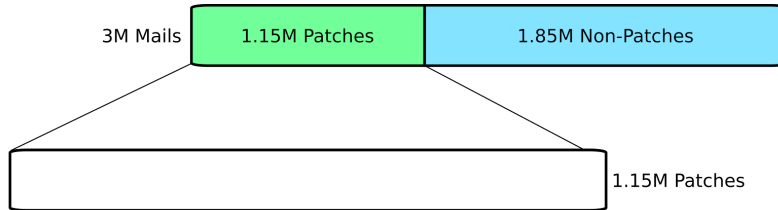
## We analyse...

- ▶ v2.6.39..linus/master
- ▶  $\approx 610K$  commits
- ▶ Mails: 2011-05-01 – 2018-12-31
- ▶  $\approx 3M$  mails
- ▶ Lists: All Public Inboxes from lore.kernel.org
- ▶ linux-amlogic, **linux-arm-kernel**, linux-i3c, linux-mtd, linux-riscv, linuxppc-dev, cocci, linux-block, linux-bluetooth, linux-btrfs, linux-cifs, linux-clk, linux-crypto, linux-ext4, linux-fsdevel, linux-hwmon, linux-iio, linux-integrity, **linux-kernel**, linux-media, linux-mips, linux-modules, linux-next, **netdev**, linux-nfs, linux-parisc, linux-pci, linux-renesas-soc, linux-rtc, linux-security-module, linux-sgx, linux-trace-devel, linux-watchdog, **linux-wireless**

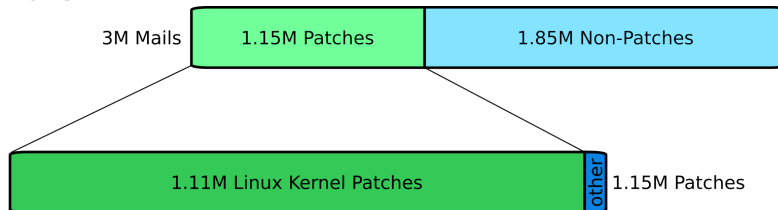
2011-05-2018-12



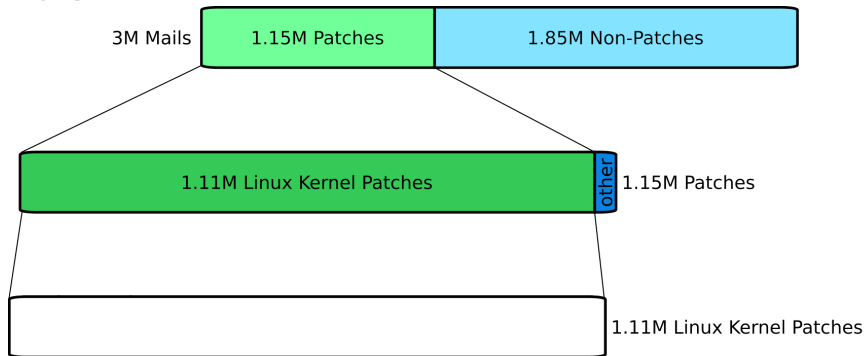
2011-05-2018-12



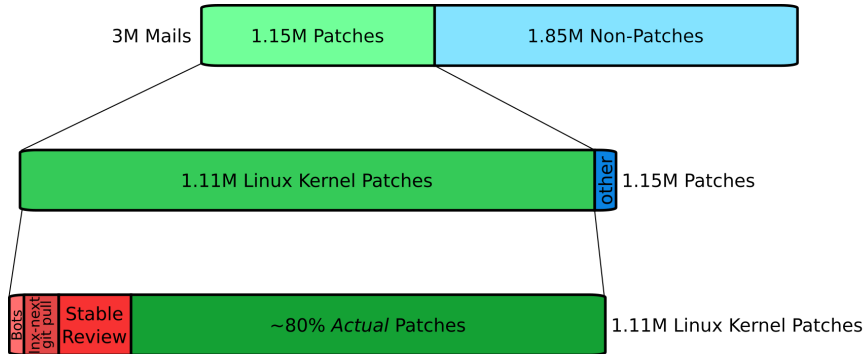
2011-05-2018-12



2011-05-2018-12

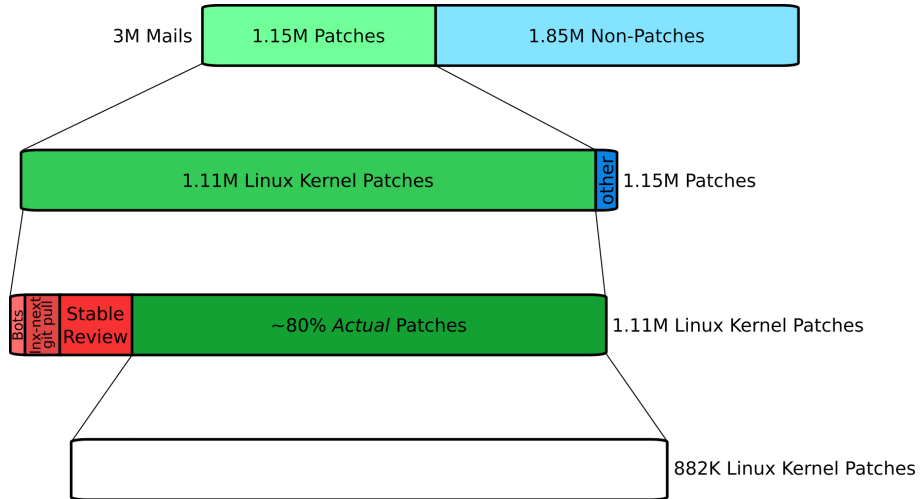


2011-05-2018-12

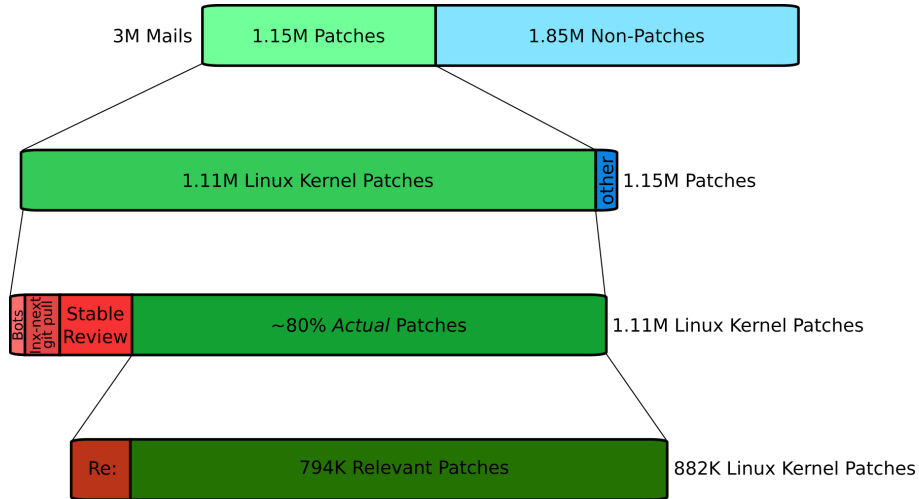




2011-05-2018-12



2011-05-2018-12



## Ignored Patches

### Research Question

Are there specific characteristics for ignored patches?

### Definition

A patch on a ML is *ignored* if...

- ▶ ... the thread of the patch has no responses from persons other than the author
- ▶ ... the patch was not accepted upstream
- ▶ ... all related patches (e.g., revisions in other series) were ignored

## Ignored Patches

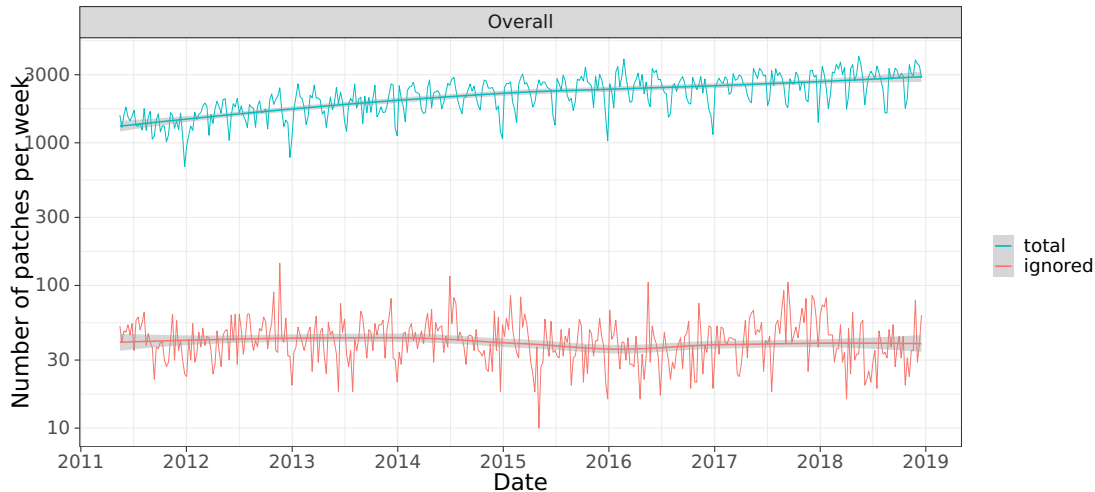
### By the Numbers...

- ▶ lore.kernel.org lists 2011-2018: ø2.5% ignored patches
  - ▶ 2011: ø3.9%
  - ▶ 2015: ø2.1%
  - ▶ 2018: ø1.6%

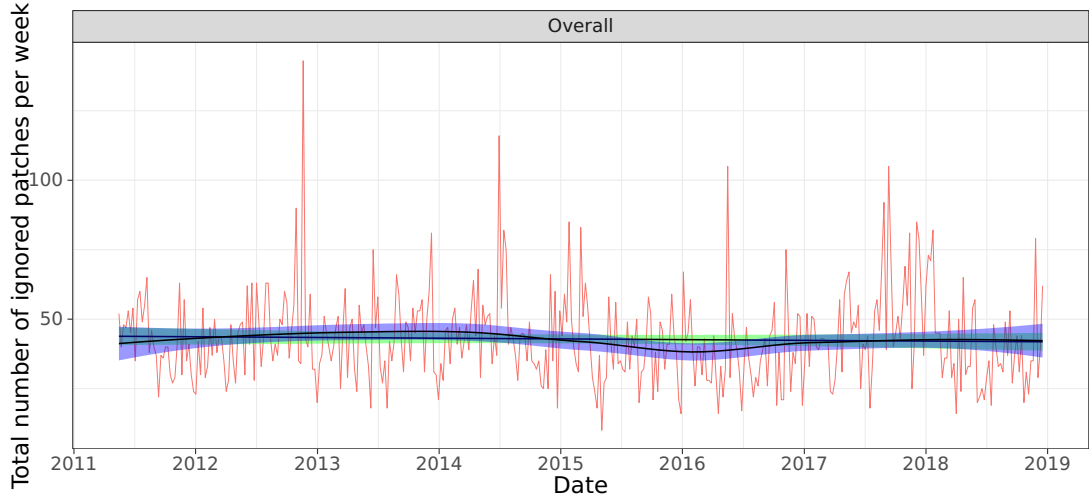
## Evolution of ignored patches



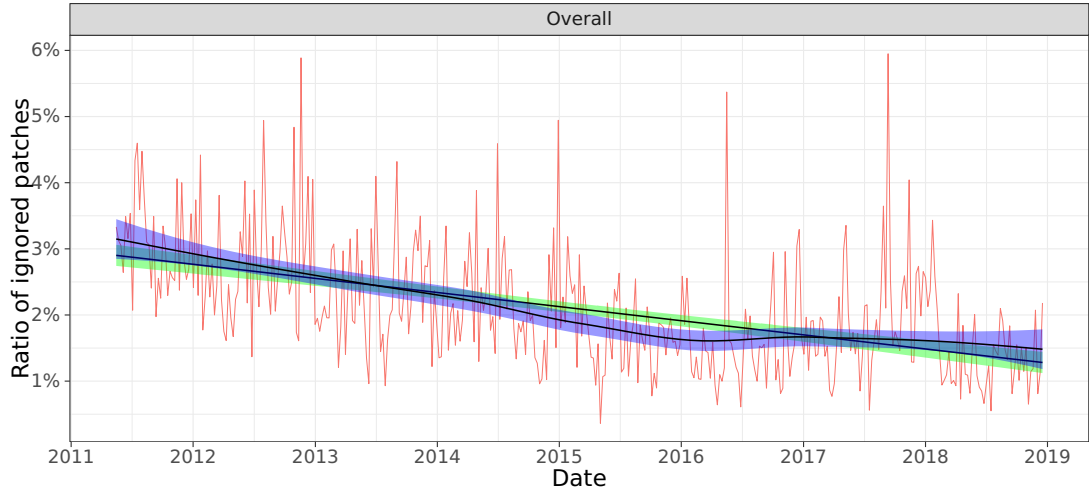
## Evolution of ignored patches



## Evolution of ignored patches

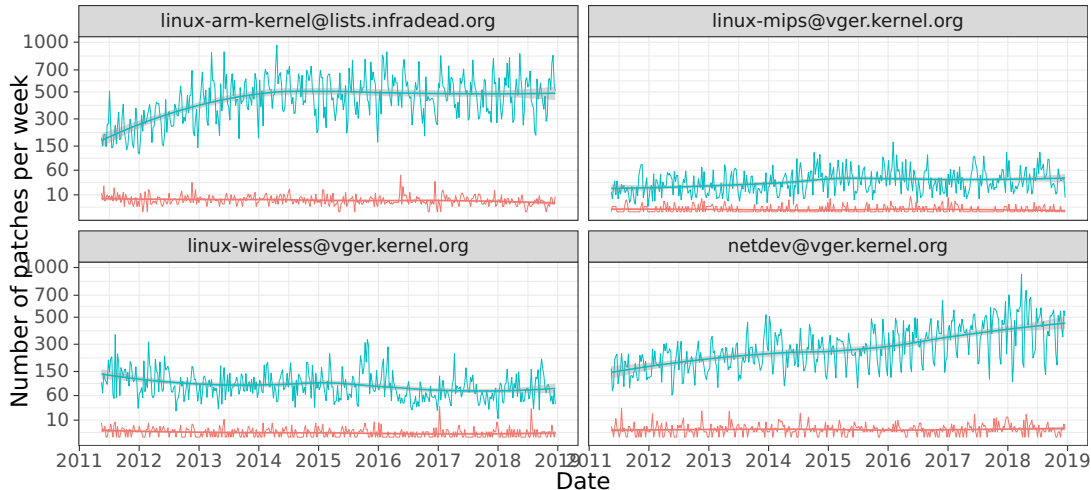


## Evolution of ignored patches

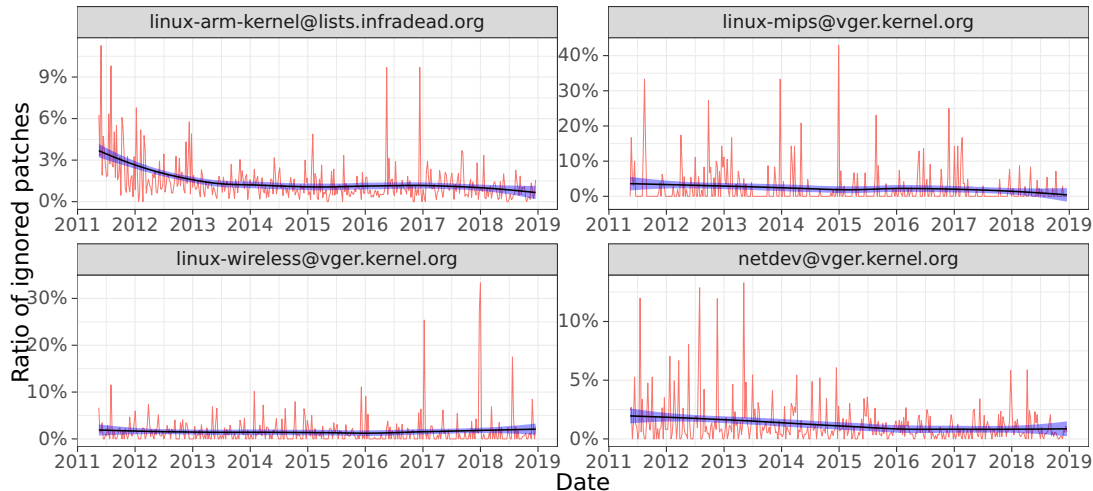




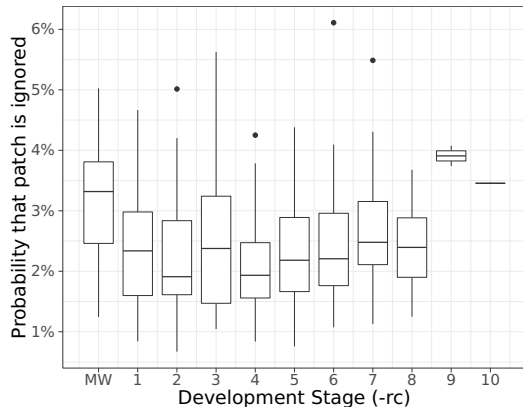
## Evolution of ignored patches



## Evolution of ignored patches



## Does it matter *when* a patch is sent?

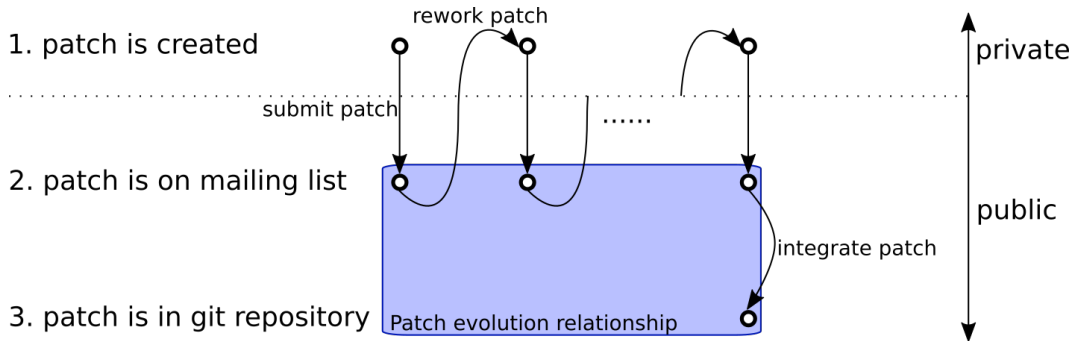


Distribution of ratio ignored/total patches grouped by linux kernel development stage

### Insights

- Largely independent of the development stage
- Slightly higher chance of ignorance during merge window

## Towards a formal model of the development process



## Off-list Patches

### Definition

An *off-list patch* is a patch that...

- ▶ ...has been included in Linus' git repository
- ▶ ...has never been sent to any public mailing list

## Off-list Patches

### Definition

An *off-list patch* is a patch that...

- ▶ ...has been included in Linus' git repository
- ▶ ...has never been sent to any public mailing list

### Results

- ▶ Identified 80 commits with PaStA heuristics from v5.1-rc1..v5.1 ( $\approx 1800$  commits)
- ▶ Manually assessed 60 commits and identified 24 off-list patch commits

## Off-list Patches

### The obvious

- ▶ Reverting patches is discussed on mailing list, the reverting patch is not sent.
- ▶ Very few patches from maintainers are actually off-list patches

### The less obvious

- ▶ Some off-list patches are clearly some security-related issues
- ▶ Patches from some subsystem maintainers are often off-list

```
commit c7084edc3f6d67750f50d4183134c4fb5712a5c8
```

```
Author: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

```
Date:   Fri Apr 5 15:39:26 2019 +0200
```

```
tty: mark Siemens R3964 line discipline as BROKEN
```

The n\_r3964 line discipline driver was written in a different time, when SMP machines were rare, and users were trusted to do the right thing. Since then, the world has moved on but not this code, it has stayed rooted in the past with its lovely hand-crafted list structures and loads of "interesting" race conditions all over the place.

After attempting to clean up most of the issues, I just gave up and am now marking the driver as BROKEN so that hopefully someone who has this hardware will show up out of the woodwork (I know you are out there!) and will help with debugging a raft of changes that I had laying around for the code, but was too afraid to commit as odds are they would break things.

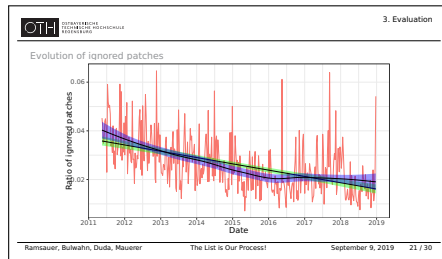
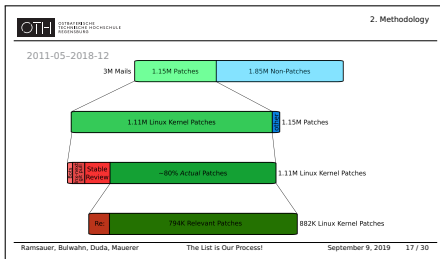
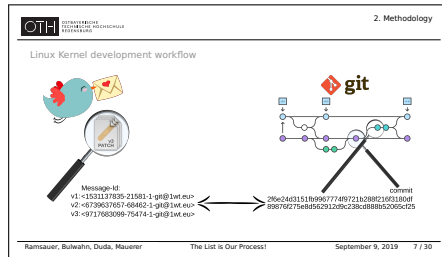
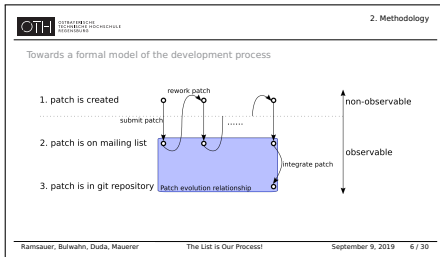
Many thanks to Jann and Linus for pointing out the initial problems in this codebase, as well as many reviews of my attempts to fix the issues. It was a case of whack-a-mole, and as you can see, the mole won.

```
Reported-by: Jann Horn <jannh@google.com>
```

```
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

```
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```





# Thank you!

`{ralf.ramsauer,wolfgang.mauerer}@othr.de`  
`sebastian.duda@fau.de`  
`lukas.bulwahn@gmail.com`